17810

RECEIVED
JUL 26 1973

Postel

DRAFT DESIGN FOR A TEXT/GRAPHICS PROTOCOL 1

This proposal should be seen as a synthesis of existing ideas rather than an attempt to put forth new ones. It is based on work by the NGG, Elaine Thomas, Peter Deutsch, Charles Irby, Ken Victor, Bill Duvall, Bob Sproull, and others at ARC, PARC, and BBN. 1a

We are concerned about the lack of text-handling capabilities of the protocol suggested in RFC 493. Also, we feel that the protocol will have a significant influence on the interface provided to writers of future graphics application programs, and consequntly that such things as "beam twiddling" should not be part of the protocol. 1b

Things of this nature address the problem at too low a level for a facility which is intended to service the needs of a wide range of graphics devices. 1b1

We feel that, although the breakdown into "levels" as proposed in RFC 493 may be expedient for initial experimentation, it is inappropriate for a Network Standard protocol. Instead, we propose that the protocal allow for two levels, segmented and structured. This allows the writers of graphics application programs to deal with a very simple display facility (segments consisting of lines, dots, or character strings) or with a powerful structure of display subroutines. 1b2

We propose an experimental implementation of such a scheme on the ARC, BBN, and PARC systems to test these ideas using several application programs (including NLS) and at least an IMLAC, ARDS, and an E&S LDS. 1c

Environment 2

We are trying to design a protocol used to communicate with a "virtual display" to operate at the other end of a wire (ARPANET connection) from a "host" which is running some kind of display application program. 2a

We will adopt the terminology that the human user, sitting at the display, is the "user" and the application program is the "server". 2a1

We wish to stress the fact that within a single application, a single terminal should be useable both as an "interactive graphics" terminal AND as an "interactive control" terminal. Thus, the graphics protocol must allow for teletype-like operations. 2b

Received at NIC July 16, 1973.

The need for two sets of connections for running graphics programs over the Net (according to our understanding) is centered about the issue of handling (being able to recover gracefully from) berserk programs (and perhaps achieving greater bandwidth through the net). 2c

We recognize this problem but also think one should be able to run graphics programs using only one set of telnet connections. Also, it seems obvious that even though one is running a graphics program, one must expect to be able to handle "unescorted" characters (not embedded in a command or response message) being sent to his terminal. 2d

Consequently, we are proposing that the graphics protocol be implemented within telnet using 8-bit BEGIN-GRAPHICS-COMMAND and END-GRAPHICS-COMMAND characters or the 8-bit transparent mode of the new telnet. This means that one will be able to run graphics programs with one, two, or more sets of telnet connections. 2e

We also strongly propose that any site which is interested in supporting display terminals for use in network graphics would be prudent to implement local control over the display (such as IGNORE-GRAPHICS-COMMANDS, RESET-TO-TTY-MODE commands from the user to the using host). Failure to take such precautions may very well lead to burned out tubes! 2f

Basic concepts 3

The Model 3a

The model we are adopting consists of an application program manipulating a (remote) display file. This file may be "segmented" or "structured", in which case it may be manipulated independently from the display itself. 3a1

For structured display files an "update display" command causes the display file to get mapped onto the display in whatever fashion is appropriate for the display. 3a1a

Part of this protocol deals with commands issued to the (remote) display file editor. This editor creates and changes the display file at the user host. 3a2

Structured Display Files 3b

A structured display file consists of named subpictures, each containing any number of named units. There are two types of units, primitive units and call units. The effect of a unit is

independent of its name or creation order within the subpicture. 3b1

Primitive units contain drawing instructions and associated coordinates that may generate visible information on the display screen. Drawing instructions and coordinates can occur only in primitive units. 3b1a

Call units give the display structure a subroutine capability. A call unit invokes the display of another subpicture. In other words, a call unit allows one subpicture to contain instances of other subpictures. As well as providing for subroutine-style control transfer, call units can be used to establish display parameters and maintain parameter transparency. For example, a call unit can be used to call a subpicture with a translation and relative intensity setting. On return from the called subpicture, these parameters are restored to their original values. 3b1b

A subpicture is an ordered list of units which can be any mixture of primitive and call units. Each subpicture begins with a header and terminates with the subpicture end unit. The subpicture end unit is a single unique unit in a display structure linked to the end of each subpicture. 3b1c

In order to understand how control passes through a structure, one can think of the display elements as follows: subpictures are subroutines and units are linked blocks of in-line code. When all of the units contained in a subpicture have been executed, the subpicture end unit returns control to wherever the subpicture was called from. A primitive unit contains display code and a transfer to the next unit. A call unit contains a subroutine call to a subpicture and a transfer to the next unit in line. 3b1d

Segmented Display Files 3c

A segmented display file consists of named segmentes, each containing any number of primitive units. The only operations available for segmented display files is to add new, delete old, or replace old segments (updating the actual display happens automatically). The effect of a unit is independent of its name or creation order within the subpicture. 3c1

Hosts 3d

Since a given terminal may be under partial control of several different hosts, all further discussion of names, coordinates,

display files, etc. should be taken as relative to each individual host. 3d

That is, each host believes it has a display file, naming, and coordinate space and a set of state parameters entirely under its control; its only evidence of resource sharing is that the terminal may appear to be of different sizes at different times. 3d2

(We feel that in principle it should be processes within hosts, rather than hosts, that enjoy these properties, but it does not seem feasible to construct a process identification scheme that all hosts will find acceptable.) 3d3

Subpictures 3e

A subpicture has a name and zero or more units. 3e1

Subpicture names are arbitrary, globally unique, fixed-length identifiers (subpicture names are chosen by the host). 3e1a

Each unit (displayable component) has a name, which is local to the subpicture. 3e1b

A unit may be a "primitive unit", such as a string or a vector, or a "call unit", which implies displaying a (possibly transformed) copy of another subpicture. 3e2

The display data are organized into a re-entrant tree (acyclic graph) by the call units. 3e2a

A unit may be "visible" or "invisible". 3e3

A particular instance of a subpicture (the result of a call-unit) appears on the screen precisely if it and all subpictures on the logical path to it from the root of the tree are "visible". 3e3a

Primitive units 3f

Strings 3f1

A string unit is a horizontal line of characters with its own mode and X,Y origin relative to the origin of the subpicture. 3f1a

Note: intensity is always relative. 3f1a1

Font and mode (e.g. blinking) information logically accompanies each character. This is accomplished by means of embedded mode and font specification characters and a "restore original string mode and font" character. 3f1b

Note: Mode modifiers are non-displayable characters and do not take up character positions on the screen. 3f1b1

Determining the space occupied on the screen by a string requires knowledge of the font(s) being used; this is a separate question which is dealt with later. 3f1c

TTY units 3f2

A tty unit is a rectangle that consists of a number of lines. Within this unit the display acts as if it were an alpha-numeric display, e.g., 3f2a

characters which would write beyond the right hand margin of the rectangle cause an automatic line folding to take place 3f2a1

ascii control characters CarriageReturn, LineFeed, FormFeed, and BackspaceCharacter, (HorizontalTab and VerticalTab?), are interpreted appropriately 3f2a2

other control characters are displayed in a terminal specific manner, e.g. ↑F, <↑F>, etc. 3f2a3

display of the characters in the range 200-377 is left unspecified at this point (truncated to 7 bits?, alternate fonts?, alternate modes?) 3f2a4

It is hoped that we can agree on a standardization of some of the characters in this range to allow for such things as greek letters, common mathamatical symbols, super-scripting, and sub-scripting. 3f2a4a

linefolding that would cause characters to be written below the rectangle (whether performed automatically or by a LineFeed character, etc.) cause the text within the unit to be scrolled upwards one line (storage tube may adopt a different scheme). 3f2a5

Characters are displayed in a teletype unit in one of two ways: 3f2b

Characters sent to the terminal that are not part of any

command (unescorted characters) are appended to appropriate tty-units (see below --- USE-TTY-UNITS, TTY) 3f2b1

By use of the APPEND-STRING-TO-UNIT command for structured display files 3f2b2

The first character sent to a tty-unit appears as the first character (at the left margin) of the top line. This is necessary for a number of reasons, the most convincing of which is the behavior characteristics of storage tubes and most real alpha-numeric displays. 3f2c

Successive characters appear as successive characters within the top line until either an explicit (e.g., linefeed) or implicit (line overflow) line break occurs. 3f2c1

When a line break occurs, the next character appears on the second from the top line of the unit. 3f2c2

This continues until the bottom line of the tty-unit is reached. 3f2c3

At this point, a line break causes the lines within the unit to scroll up one line. 3f2c3a

Note: Storage scopes may use a different technique for scrolling. 3f2c3a1

Dots 3f3

A dot unit consists of an initial X0,Y0 followed by a series of points X,Y which describe a series of dots. 3f3a

Each dot unit logically carries mode information such as blinking, relative intensity, etc. 3f3b

Lines 3f4

A line unit consists of an origin X0,Y0 followed by a series of points X,Y which describe a series of straight lines connected tail-to-head (i.e a polygon). 3f4a

Each line unit logically carries mode information such as blinking, dotted vs. solid, invisible. 3f4b

Other kinds of lines, such as conic sections, may belong in the primitive set. 3f4c

Special points 3f5

This primitive unit consists of a series of points, which will be displayed joined by lines in the best available manner. 3f5a

The intent is to use Flegal's algorithms to produce a smooth curve. 3f5b

Device-specific 3f6

This primitive unit consists of any number of device specific commands. The device type may be obtained through an intorrogation command. 3f6a

Call units 3g

In addition to the name of the referenced subpicture, a call unit may include the following transformations: 3g1

Master/instance rectangle: specifies a rectangle in the caller's space into which a specified rectangle of the callee's space is to be imaged. This provides independent scaling in each coordinate as well as translation and clipping. 3g1a

Rotation. It may be desirable to combine this with scaling using the familiar idea of homogenous transformation. 3g1b

Intensity and color control. In principle, a call could specify intensity increments (positive or negative) for each color. 3g1c

It is assumed that best effort will be used in scaling and rotation of text. We recommend replacing it by a line when all else fails. 3g1d

Initial state 3h

After the initial telnet connection is established, the first graphics command issued by the applications program should be a request for either a structured display file or for a segmented display file. 3h1

The response to this request should be whether or not the requested display file was allocated and other parameters about the virtual display, e.g. screen size, character sizes, whether or not color is available, etc. 3h1a

Before the display file is allocated, the terminal should

appear as, and simulate to the best of its ability, a Network Virtual Terminal (NVT). 3h2

Any graphic commands issued before the allocation of a display file will be ignored. 3h3

After requesting and receiving a structured display file, the following structure will exist: 3h4

There will exist a subpicture, refered to as the ICP SUBPICTURE, whose rectangular extent corresponds to the extent of the virtual display allocated to this host 3h4a

There will exist a tty-unit, refered to as the ICP TTY-UNIT, in the ICP SUBPICTURE, whose rectangular extent corresponds to the extent of the virtual display allocated to this host 3h4b

This tty-unit will consist of n lines, where n is terminal dependent and available through a query command. 3h4b1

This tty-unit will be instituted for the display of unescorted characters. 3h4b2

There will be in effect an implicit call on the ICP SUBPICTURE. 3h4c

This call is not accessible to the applications program. 3h4c1

The applications program causes the display of information by: 3h4d

1) creating primitive units in the ICP SUBPICTURE 3h4d1

2) creating call units, to created subpictures, in the ICP SUBPICTURE 3h4d2

3) using the TTY command to make visible/invisible the ICP TTY-UNIT (or change its location or size) 3h4d3

After requesting and receiving a segmented display file, the following strucure will exist: 3h5

There will exist a segment, refered to as the ICP SEGMENT. 3h5a

There will exist a tty-unit, refered to as the ICP TTY-UNIT, in the ICP SEGMENT, whose rectangular extent corresponds to the extent of the virtual display allocated to this host 3h5b

This tty-unit will consist of n lines, where n is terminal dependent and available through a query command. 3h5b1

This tty-unit will be instituted for the display of unescorted characters. 3h5b2

The applications program causes the display of information by: 3h5c

1) creating primitive units in the ICP SEGMENT 3h5c1

2) creating new segments 3h5c2

3) using the TTY command to make visible/invisible the ICP TTY-UNIT (or to relocate it or change its size) 3h5c3

Display editing primitives 4

General editing primitives 4a

REQUEST-DISPLAY-FILE(file-type) 4a1

file-type is either structured or segmented. 4a1a

This command requires a response. 4a1b

Segmented display file editing 4b

SEGMENT(Segment) 4b1

If the segment Segment already exists, then it is cleared; if it did not exist then it is created. 4b1a

Pictures are displayed within segments by the use of the primitive unit commands listed below. 4b1b

DELETE-SEGMENT(Segment) 4b2

If the segment Segment exists, then it is deleted. 4b2a

Primitive Units 4b3

All unit opperations cause immediate display on the screen. 4b3a

STRING-UNIT(Segment,Mode,X-Origin,Y-Origin,Text) 4b3b

Writes the specified string unit. 4b3b1

Mode refers to relative intensity, blinking, reverse video, color, etc. 4b3b2

Errors: Segment does not exist. 4b3b3

LINE-UNIT(Segment,Type,Mode,X0,Y0,X1,Y1, ..., Xn,Yn) 4b3c

Draws the specified line segments. 4b3c1

Type refers to solid, dashed, dotted, etc. 4b3c2

Errors: Segment does not exist; illegal mode. 4b3c3

DOT-UNIT(Segment,Mode,X0,Y0,X1,Y1, ..., Xn,Yn) 4b3d

Draws the specified dots. 4b3d1

Errors: Segment does not exist; illegal mode. 4b3d2

SPECIAL-POINTS-UNIT(Segment,Mode, X1,Y1, ..., Xn,Yn) 4b3e

Draws the special-points curve. 4b3e1

The terminal should attempt to connect the specified points in the nicest way possible (e.g. Flegal's spline curve algorithm, straight line segments). 4b3e2

Errors: Segment does not exist; illegal mode. 4b3e3

TTY-UNIT(Segment, mode, rectangle, lines) 4b3f

Creates a unit which will behave as a tty-simulation area with "lines" lines distributed within the specified rectangle. 4b3f1

Unescorted characters will be echoed in this unit in addition to any other units they are being sent to. 4b3f2

Errors: Segment does not exist. 4b3f3

DEVICE-SPECIFIC-UNIT(Segment,device commands) 4b3g

Creates a unit of device specific commands. 4b3g1

TTY(parameters) 4b4

parameters are: 4b4a

position rectangle, visible/invisible, number of lines, mode of characters 4b4a1

This refers to the ICP TTY simulation. 4b4b

RESET() 4b5

deletes all segments, except ICP SEGMENT, and all units of ICP SEGMENT, except ICP TTY-UNIT 4b5a

resets all modes to their initial state (i.e., the state that existed immediately after a REQUEST-DISPLAY-FILE command) 4b5b

Structured display file editing 4c

SUBPICTURE(Subpicture, rectangle) 4c1

Creates a new subpicture with name "Subpicture". "rectangle" is the cordinates of a diagonal of the subpicture's virtual screen (i.e. its coordinate system). 4c1a

If a subpicture named "Subpicture" already exists, it is cleared and the new coordinate rectangle takes precedence. 4c1b

DELETE-SUBPICTURE(Subpicture) 4c2

Deletes the subpicture named Subpicture. Call units referring to Subpicture are also deleted. 4c2a

CLEAR-SUBPICTURE(Subpicture) 4c3

Deletes all units of the subpicture Subpicture, but does not delete the subpicture. 4c3a

Primitive Units 4c4

All the operations for creating units are transparent to the prior existence of the designated unit, i.e. they function as "replace" as well as "create". 4c4a

STRING-UNIT(Subpicture,Unit,Target-Key,Mode,X-Origin,Y-Origin,Text) 4c4a1

Replaces the unit by a string unit. 4c4a1a

Mode specifies the mode of the characters (e.g. blinking, underlined, etc). 4c4a1b

Target-Key is used in conjunction with the TARGET-SENSITIVE command and target input. It may also be set via the SET-TARGET-KEY COMMAND. 4c4a1c

Errors: Subpicture does not exist; X-Origin or Y-origin is outside the subpicture's virtual coordinate system. 4c4a1d

We explicitly do not require an error if the string extends beyond the right-hand edge of the subpicture; however, the results are not defined. 4c4a1d1

LINE-UNIT(Subpicture,Unit,Target-Key,Type,Mode,X0,Y0,X1,Y1, ..., Xn,Yn) 4c4a2

Replaces the unit by a line unit. 4c4a2a

Errors: Subpicture does not exist; illegal mode; some X or Y is outside the subpicture. 4c4a2b

DOT-UNIT(Subpicture,Unit,Target-Key,Mode,X0,Y0,X1,Y1, ..., Xn,Yn) 4c4a3

Replaces the unit by a dot unit. 4c4a3a

Errors: Subpicture does not exist; illegal mode; some X or Y is outside the subpicture. 4c4a3b

SPECIAL-POINTS-UNIT(Subpicture,Unit,Target-Key, Mode, X1,Y1, ..., Xn,Yn) 4c4a4

Replaces the unit by a special-points unit. 4c4a4a

Errors: Subpicture does not exist; illegal mode; some X or Y is outside the subpicture. 4c4a4b

CALL-UNIT(Subpicture,Unit,Target-Key,Called-Subpicture,Parameters) 4c4a5

Replaces the unit by a call unit. 4c4a5a

Parameters: 4c4a5b

Master-Instance rectangles 4c4a5b1

rotation 4c4a5b2

mode 4c4a5b3

Errors: Subpicture does not exist; Called-Subpicture does not exist; parameter errors. 4c4a5c

TTY-UNIT(Subpicture, unit, mode, rectangle, lines) 4c4a6

Creates a unit which will behave as a tty-simulation area with "lines" lines distributed within the specified rectangle. 4c4a6a

Errors: Subpicture does not exist. 4c4a6b

DEVICE-SPECIFIC-UNIT(Subpicture,Unit,Target-Key,device commands) 4c4a7

Creates a unit of device specific commands. The action of the commands should leave alone (or at least restore) any global modes, e.g., the standout mode (see below). 4c4a7a

APPEND-STRING-TO-UNIT(Subpicture, unit, Text) 4c5

Appends the specified text to the specified unit. This only makes sense if the specified unit is a string or tty unit. 4c5a

Errors: Subpicture does not exist, unit does not exist, not a string or tty unit. 4c5b

DELETE-UNIT(Subpicture,Unit) 4c6

Deletes a unit. 4c6a

VISIBLE-UNIT(Subpicture, Unit,Flag) 4c7

Makes the Unit visible or invisible as specified by Flag. 4c7a

If a unit which is target sensitive is made invisible, it is no longer target sensitive. However, in the absense of a subsequent modifying target sensitive command, the unit becomes target sensitive again if it should be made visible. 4c7b

Errors: Subpicture does not exist, unit does not exist. 4c7c

SET-TARGET-KEY(Subpicture, Unit, Target-Key) 4c8

sets the target key for the specified unit to the specified value. 4c8a

SET-STANDOUT-MODE(mode) 4c9

Sets the mode that will be used to make text and/or units stand out to blinking, underlining, etc. 4c9a

If the terminal does not support the specified mode, the terminal should make a best effort or use another method to make things stand out. 4c9b

STANDOUT-UNIT(Subpicture, unit, yesno) 4c10

makes the specified unit stand out (according to the mode set by SET-STANDOUT-MODE) or not, according to "yesno". If the unit which is to stand out is a call-unit, the instance of the subpicture which is the result of the call (all the way to the terminal nodes) is made to stand out. 4c10a

STANDOUT-TEXT(Subpicture, unit, begin-char-count, end-char-count, yesno) 4c11

Unit must refer to a string unit. 4c11a

Makes the specified text stand out (according to the mode set by SET-STANDOUT-MODE) or not, according to "yesno". 4c11b

UPDATE-STRUCTURED-DISPLAY() 4c12

This causes any changes that have been made to the display file, since the last update or since ICP, to be reflected on the screen. 4c12a

TTY(parameters) 4c13

parameters are: 4c13a

position rectangle, visible/invisible, number of lines, mode of characters 4c13a1

This refers to the ICP TTY simulation. 4c13b

USE-TTY-UNITS(Subpicture1, unit1, ..., Subpicturen, unitn) 4c14

Unescorted characters are to be appended only to the specified tty units. 4c14a

Errors: Subpicture,unit does not exist. 4c14b

RESET(How) 4c15

Case How Of 4c15a

= Permanent 4c15a1

Immediately resets the terminal to its initial ICP state 4c15a1a

= Temporary 4c15a2

Immediately resets the terminal to its initial ICP state without destroying the previous state. 4c15a2a

= Restore state saved from last RESET(Temporary). 4c15a3

Direct Feedback 5

It seems extremely desireable, given network speeds, to allow the using host to perform direct feedback to the user without intervention from the application program in the serving host. This is already done in telnet with local echoing. We propose extending this capability to graphics by allowing "dragging" (attaching a subpicture's origin to the position of the cursor), "tracking" (following the movement of the mouse, stylus, or light pen with a distinctive mark on the screen), "inking" (plotting the trail of the cursor on the screen) and "rubber banding" (a straight line attached to a fixed point on one end and the cursor location on the other). 5a

These should be seen as allowable extensions of the protocol rather than as requirements. There should, however, be commands available in the protocol for determining their existence and controlling them. 5b

Data input primitives 6

Input control 6a

TARGET-SENSATIVE(key1, ..., keyn) 6a1

Arms the units which have the specified keys for target selection. 6a1a

SET-INPUT-MODE(Device, parameters) 6a2

Selects the mode in which a logical device shall produce input and under what conditions. 6a2a

the logical devices are specified below as well as their possible input formats and conditions. 6a2b

Errors: no such device. 6a2c

Keyboard input 6b

The keyboard has only one input mode, in which it sends a character whenever a key is struck. 6b1

Binary devices 6c

Unless otherwise specified, binary devices act as an extension of the keyboard and produce 8-bit characters which are not distinguishable from keyboard characters by the serving host. 6c1

The algorithm for translating binary devices into characters is not specified, but something like the NLS accumulation algorithm for mouse-keyset chords is intended. 6c1a

Binary devices may also input binary data (according to their up/down states), which is transmitted on state changes. Examples of this type of device are function keys and overlay cards, mouse and keyset (used independently or together), pen-up/pen/down, light pen buttons, etc. 6c2

Coordinate input 6d

Coordinates may be sent according to any subset of the following criteria: With every character in some designated set (e.g. control characters, or all characters); with every binary device state change input; after some time interval has elapsed; after a position change P > (y1-y0)↑2+(x1-x0)↑2, etc. 6d1

Coordinates may be sent in either or both of "X-Y" or "target" format. 6d2

X-Y format is just the location of the cursor relative to the screen region assigned to the host. 6d2a

Target format is the "call stack" (logical path from the root unit - the ICP SUBPICTURE - to the closest unit) plus the target-key of that unit plus the count of the closest character within the string or the closest line segment or dot or special point if appropriate. 6d2b

Target input is unavailable for segmented display files. 6d2b1

In the event of overlapping target sensitive units, it is not specified which of the units selected will be returned as the hit unit. 6d2b2

Time input 6e

Since hosts may wish to consider two events happening sufficiently close together to be simultaneous, or to keep detailed interaction statistics, it must be possible to request time information to be sent with some reasonable subset of other types of input. 6e1

Interrogations 7

It must be possible for the serving host to discover its environment (e.g. screen size, available devices) and to read back state information (display file). 7a

This is very desireable both for debugging and for redirecting a displayed image to another device (e.g. a plotter). 7a1

Environment 7b

Terminal parameters: screen size and resolution, available input devices, terminal type (for device specific control), number of lines in the ICP TTY-UNIT. 7b1

Character parameters: available character sizes, special (non-ASCII) characters, font characteristics, sub- and super-scripting facilities. 7b2

State 7c

Display file or display file components. 7c1

Cursor Position 7d

It should be possible for the application program to read the cursor position at any time. 7d1

Display File Support 7e

It should be possible to find out if this user process supports only segmented or structured display files, or both. 7e1

Command support 7f

It should be possible to get a matrix from the user process which indicates which commands are implemented. This is a neccessity to find out which, if any, of the direct feedback features are supported, and might be nice to allow for, e.g., the possibility of a text only or grapgics only subset of the protocol to be implemented. 7f1

Encoding Principles 8

Commands will have the format: BGC OPCODE DATA EGC where: 8a

BGC (Begin Graphics Command) places the telnet connection into a "read graphics command" mode, 8a1

OPCODE DATA is the specific graphics command and data, and 8a2

EGC (End Graphics Command) restores the telnet connection to its normal state. 8a3

Note: This may all have to be bracketed by telnet Begin-8-bit-transparent-mode and End-8-bit-transparent-mode commands. 8a4

Numbers in general will have have 7-bits of significance in each byte -- if the high order bit of a byte is on, then the significant bits from the next byte should be concatenated onto the low-order end of the bits collected so far, etc.. 8b

Subpicture names - shall be 14-bit numbers, assigned by the serving host. 8c

Unit names - shall be 14-bit numbers, assigned by the serving host. 8d

Strings - shall be 8-bit characters, with an escape convention to represent changes of font and mode. 8e

Since the channel is 8 bits wide, there is room for many more than 128 displayable characters. However, the interpretation of codes 200B and above is not standardized! 8e1

Coordinates should be as described in RFC 493. 8f

Rectangles - shall be specified by the coordinates of the endpoints of one of the diagonals 8g

Encoding 9

The actual encoding of this protocol is forthcoming. Since we expect some changes to come about because of the upcomming Network Graphics Group Meeting, we have postponed the actual encoding until after this meeting. 9a